APPLICATION FOR PATENT

Inventors: Amir Averbuch and Yossi Keller

Title: **A method for fast motion estimation using bi-directional and symmetrical gradient schemes**

FIELD AND BACKGROUND OF THE INVENTION

Moving objects are often characterized by a coherent motion (rotation and translation) that is distinct from that of the background. This makes motion a very useful feature for segmenting video sequences. It can complement other features such as color, intensity, or edges that are commonly used for segmentation of still images.

We start by defining the term motion. We denote by I(x,y;k) the intensity or luminance of pixel (x,y) in frame k. Following the definitions in A.M. Tekalp, Ed., "Digital Video Processing", Upper Saddle River, NJ, Prentice-Hall, 1995 (hereinafter TEK95), we have to distinguish between two-dimensional (2-D) and apparent motion. The projection of the three-dimensional (3-D) motion onto the image plane is referred to as 2-D motion. It is the true motion that we would like to automatically detect. On the other hand, apparent motion is what we perceive as motion and is induced by temporal changes in the image intensity I(x,y;k). Apparent motion can be characterized by a correspondence vector field or by an optical flow field. A correspondence vector describes the displacement of a pixel between two frames, whereas the optical flow (u,v) at pixel (x,y;k) refers to a velocity and is defined as

$$(u,v)=\left(\frac{\partial x}{\partial t},\frac{\partial y}{\partial t}\right) \tag{1}$$

The optical flow and correspondence vectors are related. From Eq. 1 it can also be seen that apparent motion is highly sensitive to noise, which can cause largely incorrect results. Further, moving objects or regions must contain sufficient texture to generate optical flow, because the luminance in the interior of moving regions with

uniform intensity remains constant. Unfortunately, we can only observe apparent motion.

Motion estimation

Besides the difficulties already mentioned, motion estimation algorithms have to solve the so-called occlusion and aperture problem. The occlusion problem refers to the fact that no correspondence vectors exist for covered and uncovered background. To illustrate the aperture problem, we fist introduce the optical flow constraint (OFC). It assumed that the intensity remains constant along the motion trajectory (TEK95), i.e.,

$$\begin{aligned}\frac{\partial}{\partial t}I(x,y;k) &= \frac{\partial I}{\partial x}\cdot\frac{\partial x}{\partial t}+\frac{\partial I}{\partial y}\cdot\frac{\partial y}{\partial t}+\frac{\partial I}{\partial t}\\ &= \langle \nabla I,(u,v)\rangle+\frac{\partial I}{\partial t}=0\end{aligned} \tag{2}$$

where $\langle\cdot,\cdot\rangle$ denotes the vector inner product. The aperture problem states that the number of unknowns is larger than the number of observations. From the optical flow constraint (Eq. 2) it follows that only the flow component in the direction of the gradient, the so-called normal flow, can be estimated. The orthogonal component can take on any value without changing the inner product, and is therefore not defined. Thus, additional assumptions are necessary to obtain a unique solution. These usually impose some smoothness constraints on the optical flow field to achieve continuity.

There are two ways of describing motion fields:

1. Nonparametric representation. A dense field is estimated where each pixel is assigned a correspondence or flow vector. Then block matching is applied. The current frame is subdivided into blocks of equal size, and for each block the best match in the next (or previous) frame is computed. All pixels of a block are assumed to undergo the same translation, and are assigned the same correspondence vector. The selection of the block size is crucial. Block matching is unable to cope with rotations and deformations. Nevertheless, the simplicity and relative robustness of block matching makes it a popular technique. Nonparametric representations are not suitable for segmentation because an object moving in the 3-D space generates a spatially varying 2-D motion field even with the same region, except for the simple

case of pure translation. This is the reason why parametric models are commonly used in segmentation algorithms. However, dense field estimation is often the first step in calculating the model parameters.

2. Parametric models require a segmentation of the scene, which is our ultimate goal, and describe the motion of each region by a set of a few parameters. The motion vectors can then be synthesized from these model parameters. A parametric representation is more compact than a dense field description and less sensitive to noise, because many pixels are treated jointly to estimate a few parameters.

In order to derive a model or transformation that describes the motion of pixels between successive frames, assumptions on the scene and objects have to be made. Let (X,Y,Z) and (X',Y',Z') denote the 3-D coordinates of an object point in frame k and k+1, respectively. The corresponding image plane coordinates are (x,y) and (x',y'). If a 3-D object undergoes translation, rotation and linear deformation, the 3-D displacement of a point on the object is given in (G. Wolberg, "Digital Image Warping", IEEE, 1984, hereinafter WOL84)

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \\ s_{31} & s_{32} & s_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \tag{3}$$

It is very common to model 3-D objects by (piecewise) planar patches whose points satisfy

$$aX+bY+cZ=1. \tag{4}$$

If such a planar object is moving according to Eq. 3, one obtains the "affine motion model" under orthographic projection and the "eight-parameter model" under perspective projection.

The 3-D coordinates are related to the image plane coordinates under the orthographic (parallel) projection by

$$(x,y) = (X,Y) \quad \text{and} \quad (x',y') = (X',Y'). \tag{5}$$

This projection is computationally efficient and is good approximation if the distance between the objects and the camera is large compared to the depth of the objects. From Eqs. 3-5, it follows that

$$\begin{aligned} x' &= a_1 x + a_2 y + a_3 \\ y' &= a_4 x + a_5 y + a_6 \end{aligned} \tag{6}$$

which is known as the affine model. In the case of the more realistic perspective (central) projection, one gets

$$(x,y) = \left(\frac{X}{Z}, \frac{Y}{Z}\right) \text{ and } (x',y') = \left(\frac{X'}{Z'}, \frac{Y'}{Z'}\right) \tag{7}$$

Together with Eqs. 3 and 4, this results in the eight-parameter model

$$\begin{aligned} x' &= \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1} \\ y' &= \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1} \end{aligned} \tag{8}$$

Both the affine and the eight-parameter models are very popular, but many other transformations exist depending on the assumption made.

Parametric models describe each region by one set of parameters that is either estimated by fitting a model (in the least squares sense) to a dense motion field obtained by a nonparametric method, or directly from the luminance signal I(x,y;k) as in M. Hotter and R. Thoma, "Image segmentation based on object oriented mapping parameter estimation", Signal Processing, vol. 15, no. 3, pp. 315-334, 1988 (hereinafter HOT88), and in H. G. Musmann, M. Hotter, and J. Ostermann, "Object-oriented analysis-synthesis coding of moving images", Signal Processing: Image Commun., vol. 1, pp. 117-138, 1989 (hereinafter MUS89). Although parametric representations are less noise sensitive, they still suffer from the intrinsic problems of motion estimation. It should be noticed that one has to be careful when interpreting an estimated flow field. Most likely, it is necessary to include additional information such as color or intensity to accurately and reliably detect boundaries of moving objects.

Image registration plays a vital role in many image processing applications such as video compression (TEK95; F. Dufaux and J. Konrad, "Efficient, Robust, and Fast Global Motion Estimation for Video Coding", IEEE Transactions on Image Processing, vol. 9, no. 3, pp. 497-501, March 2000 (hereinafter DUF00)), video enhancement (M. Irani and S. Peleg, "Motion Analysis for Image Enhancement: Resolution, Occlusion, and Transparency", Journal of Visual Communication and Image Representation, Vol. 4, No. 4, pp. 324-335, December 1993 (hereinafter IRA93)) and scene representation (S. Man and R. W. Picard, "Virtual Bellows: constructing high quality stills from video", Proc. IEEE Int. Conf. Image Processing Austin, TX, Nov. 13-16, 1994 (hereinafter MAN94), R. Szeliski, "Image Mosaicing for Tele-Reality Applications", CRL 94/2, May 1994 (hereinafter SZE94), and M. Irani, P. Anandan, and S. Hsu, "Mosaic based representations of video sequences and their applications", International Conference on Computer Vision, pages 605-611, 1995 (hereinafter IRA95)). It has drawn a significant research attention. A comprehensive comparative survey by L. Barron, D. J. Fleet and S. S. Beauchemin, "Performance of Optical flow Techniques", Int. J. Computational Vision, Vol. 12, pp. 43-77, 1994 (hereinafter BAR94) found the family of gradient-based motion (GM) estimation methods, originally proposed by B. K. P. Horn and B. G. Schunck, "Determining Optical Flow", Artificial Inteligence, vol. 17, pp. 185-203, 1981 (hereinafter HOR81), to perform especially well. The purpose of the GM algorithm is to estimate the parameters vector $\underline{P}$ associated with the *parametric image registration* problem: starting from pure global translation (2 parameters), image plane rotation (4 parameters), 2-D affine (6 parameters), and pseudo-projective (8-parameter flow).

These models have been used extensively and are estimated directly from image spatio-temporal derivatives using coarse-to-fine estimation via Gaussian pyramids (multi-scale). These methods search for the best parametric geometric transform that minimizes the square of change between image intensities (SSD) over the whole image. An updated comprehensive description of these methods can be found in (M. Irani and P. Anandan, "All About Direct Methods", ftp://ftp.robots.ox.ac.uk/pub/outgoing/phst/summary.ps.gz (hereinafter IRAftp)).

Let $I_1(x,y)$ and $I_2(x,y)$ be two images **20** and **22** respectively that have some common overlap as shown by a shaded area **24** in FIG. 1. Then each pixel in their common support (area **24**) satisfies:

$$I_1(x,y) = I_2\left(\tilde{x}(x,y,\underline{P}), \tilde{y}(x,y,\underline{P})\right) \quad (9)$$

where $\underline{P}$ is a parameters vector having a structure that depends on the type of the estimated motion model. Gathering and solving all the equations associated with pixels in the mutual support (area **24**) ($\underline{P}$ is assumed to be constant over the whole mutual area **24**), estimates the ***global motion*** between the images (SZE94), thus gaining robustness (in the estimated parameters of the global motion estimation) due to very highly over-constrained linear systems (each pixel contributes a linear constraint). Gathering the equations related to small image patches estimates ***local motion*** (B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision"', in Proc. DARPA, Image Understanding Workshop, pp. 121-130, 1981 (hereinafter LUC81)). Equation 9 is preferably solved using non-linear iterative optimization techniques such as Gauss-Newton (W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, "Numerical Recipes in C: The Art of Scientific Computing", Cambridge Univ. Press, 1988 (hereinafter PRE88)) and Levenberg-Marquardt (LM) (SZE94) described in section 4.1. A critical implementation issue concerning the GM is the convergence range and the rate of convergence while estimating large image motions: as the estimated motion becomes larger, the convergence rate decreases, and the GM may diverge to a local minima. A possible solution is to bootstrap the motion estimation process with a different motion estimation algorithm (C. D. Kuglin and D. C. Hines. "The phase correlation image alignment method", IEEE Conference on Cybernetics and Society, pp. 163-165, September 1975 (hereinafter KUG75), and DUF00), which is robust to large motions.

Gradient method based motion estimation

The GM methodology (IRAftp) estimates the motion parameters vector $\underline{P}$ by minimizing the intensity discrepancies between the images $I_1(x,y)$ and $I_2(x,y)$ described in FIG. 1

$$\underline{P}^* = \arg\min_{\underline{P}} \left\{ \sum_{(x_i,y_i)\in s} \left(I_1\left(x_i^{(1)}, y_i^{(1)}\right) - I_2\left(f\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right), g\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right)\right)^2 \right) \right\} \quad (10)$$

where

$$x_i^{(2)} = f\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right)$$
$$y_i^{(2)} = g\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right) \tag{11}$$

and S (in Eq. 10) is the set of coordinates of pixels that are common to I_1 and I_2 in I_1's coordinates.

The formulation of MAN94 and SZE94 is followed by solving Eq. 10 using non-linear iterative optimization techniques such as Gauss-Newton and Levenberg-Marquardt, cited above. The basic GM formulation and the iterative refinement stage, as well as their embedding in a multi-resolution scheme are described next.

Basic GM formulation

The non-linear optimization of Eq. 10, is conducted via a linearization procedure, which is based on a pixel-wise, first order Taylor expansion of I_1 in terms of I_2, as a function of the parameters vector $\underline{P}$:

$$I_1\left(x_i^{(1)}, y_i^{(1)}\right) = I_2\left(x_i^{(2)}, y_i^{(2)}\right) + \sum_{P_i \in \underline{P}} \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_i} P_i \tag{12}$$

where $\frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_i}$ is the partial spatial derivative, $(x_i^{(1)}, y_i^{(1)})$ and $(x_i^{(2)}, y_i^{(2)})$ are the i^{th} common pixel in I_1 and I_2, respectively. By gathering the pixel-wise equations, a system

$$\underline{\underline{H}}\,\underline{P} = \underline{I_t} \tag{13}$$

is formulated, where

$$\begin{aligned} \underline{\underline{H}}_{i,j} &= \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_j} \\ &= \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial x_2} \frac{\partial f\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right)}{\partial P_j} + \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial y_2} \frac{\partial g\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right)}{\partial P_j} \end{aligned} \tag{13'}$$

and

$$\underline{I_t}_i = I_1\left(x_i^{(1)}, y_i^{(1)}\right) - I_2\left(x_i^{(2)}, y_i^{(2)}\right). \tag{14}$$

Equation 13` is formulated using the chain rule. Equation 13 can be solved using regular least square (PRE88):

$$\underline{P} = \left(\underline{\underline{H}}^t \underline{\underline{H}}\right)^{-1} \underline{\underline{H}}^t \underline{I_t} \tag{15}$$

where $\underline{H}^t$ is the transpose of $\underline{H}$. The expressions for $\underline{H}^t\underline{H}$ and $\underline{H}^t\underline{I_t}$ can be derived analytically by direct calculation:

$$\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)_{k,j} = \sum_i \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_k} \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_j} \tag{16}$$

$$\left(\underline{\underline{H}}^t \underline{I_t}_i\right)_k = \sum_i \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_k} \underline{I_t}_i \tag{17}$$

Algorithm flow

FIG. 2 shows a flow chart of the basic steps in the implementation of the GM algorithm. The flow chart includes a single iteration module or "phase" **30** reflecting the basic GM iteration, comprised of a computation step **32** and a solution step **34**. The basic GM iteration is typically as follows:

1. The matrix $\underline{H}^t\underline{H}$ and vector $\underline{H}^t\underline{I_t}$ are computed in step **32** using Eqs. 16 and 17, respectively.
2. Eq. 15 is solved in step **34** using singular value decomposition (PRE88).
3. The GM returns $\underline{P}$ as its output (result) in an output step **38**.

The single iteration phase is incorporated in an iterative solution, given below.

Iterative solution of the GM

Denote:

$\underline{P}_0$ - an initial estimated solution of Eq. 10 given as input, such that $Warp(I_2, \underline{P}_0) \approx I_1$

$\underline{P}_n$ - the estimated solution after n=1,...iterations

Then, the n^{th} iteration of the motion estimation algorithm becomes (FIG. 2):

1. The input image I_2 is warped towards I_1 in a warping step **36** using the current estimate $\underline{P}_n$ and it is stored in $\tilde{I}_2$ for $n > 0$. For n=0, $\underline{P}_0$ is given as an input.
2. I_1 and $\tilde{I}_2$ are used as input images to the procedure described in the basic GM formulation section above.
3. The result of the previous - $\underline{\Delta P}$, is used to update the solution in step **38**:

$$\underline{P}_{n+1} = \underline{\Delta P} + \underline{P}_n \qquad n \geq 0$$

4. Use a checking step **40** and go back to step 1 above until one of the following stopping criteria is met:
   a. At most n_{max} iterations are performed

   or

   b. The process is stopped if the translation parameters within the updated term $\underline{\Delta P}$ reach a predetermined threshold.

Gradient methods with multi-scale scheme

In order to improve the robustness and reduce the complexity of the algorithm, the iterative process is embedded in a coarse-to-fine multi-scale formulation. The robustness analysis is given in the Multiresolution GM scheme section below. The coarse-to-fine formulation is described next. A thorough description can also be found in MAN94:

1. The input images I_1 and I_2 are smoothed. Experience shows that a separable averaging filter is suitable for this task.

2. The input images I_1 and I_2 are down-sampled through multi-scale decomposition, until a minimal size of their mutual area is reached. The minimal mutual area size depends upon the estimated motion model, while the resolution step depends upon the motion estimation accuracy at each resolution level.

3. Starting with the coarsest scale, the initial estimate $\underline{P}_0$ is used to bootstrap the iterative refinement algorithm described in the iterative solution of the gradient methods section

4. The result of the iterative refinement from coarse-to-fine (step 2) is recursively repeated until the original image size is reached.

Convergence analysis of gradient methods

In order to analyze the convergence properties of the GM algorithm, the convergence properties of the general Gauss-Newton algorithm are examined next.

General convergence analysis of the Gauss-Newton algorithm

The convergence of the Gauss-Newton algorithm can be divided into two distinct phases:

$$\|\varepsilon_{k+1}\| \le C_1 \|\varepsilon_k\| + C_2 \|\varepsilon_k\|^2 \tag{18}$$

where:

$$C_1 = \left\| \left[A(\underline{x}_k) A^T(\underline{x}_k) \right]^{-1} \sum_{n=1}^{m} \frac{\partial r_n^2(x_k)}{\partial x} r_n(\underline{x}_k) \right\|$$
$$C_2 = \left\| \left[A(\underline{x}_k) A^T(\underline{x}_k) \right]^{-1} \right\| \tag{19}$$

ε_k is the parameters estimation error after iteration k

$r_n(\underline{x}_k)$ is the error associated with the n^{th} equation at iteration k

$A(\underline{x}_k)$ is the Jacobian matrix at iteration k.

By rearranging the GM formulations developed in the previous sections, these expressions are interpreted in the context of the GM formulation. C_1 and C_2 will be denoted C_1^{GM} and C_2^{GM}, respectively.

$$r_n(\underline{x}_k) = I_1\left(x_i^{(1)}, y_i^{(1)}\right) - I_2\left(f\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right), g\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right)\right) \tag{20}$$

where the parameters vector $\underline{x}$ identifies with $\underline{P}$:

$$A_{i,k} = \frac{\partial r_n(\underline{x}_k)}{\partial x_k} = \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_k} \tag{21}$$

and the equation index n is identified with the GM index i, since there is one equation per common pixel. Thus, $A(\underline{x}_k)$ is identified with the matrix $\underline{\underline{H}}$ defined in Eq. 13 and the second derivative $\frac{\partial^2 r_n(\underline{x}_k)}{\partial x_k^2}$ is given by:

$$\frac{\partial^2 r_n(\underline{x}_k)}{\partial x_k^2} = \frac{\partial^2 I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_k^2}. \tag{22}$$

Therefore, we have

$$C_1^{GM} = \left\| \left[\frac{\partial I_2}{\partial \underline{P}} \frac{\partial I_2}{\partial \underline{P}}^T \right]^{-1} \sum_{n=1}^{m} \frac{\partial^2 I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_k^2} r_n(\underline{x}_k) \right\| \tag{23}$$

$$C_2^{GM} = \left\| \left[\frac{\partial I_2}{\partial \underline{P}} \frac{\partial I_2}{\partial \underline{P}}^T \right]^{-1} \right\|. \tag{24}$$

The basic GM equation (Eq. 12) is solved by the Gauss-Newton algorithm using the LS formulation given in Eq. 20. The error associated with each equation is the truncation error of the first order Taylor expansion (PRE88):

$$\varepsilon_{GM} = \varepsilon\left(x_i^{(2)}, y_i^{(2)}, \underline{P}\right) = \frac{1}{2}\sum_{P_i \in \underline{P}} \frac{\partial^2 I_2(x_i^{(2)}, y_i^{(2)})}{\partial P_i^2}\left(P_i - P_i^*\right)^2 \qquad (25)$$

where $\underline{P}^*$ is the optimal solution (parameters vector) for the optimization problem.

In the GM setup, the sum of second partial derivatives $\frac{\partial^2 I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_i^2}$ does not strongly depend on the motion parameters vector $\underline{P}$, and the magnitude of C_1^{GM} is dominated by $\left\|\underline{P} - \underline{P}^*\right\|$. Hence, for large motions $\left\|\underline{P} - \underline{P}^*\right\| \gg 0$, $C_1^{GM} \gg 0$ and the error decay rate becomes linear rather than quadratic. Therefore, the convergence of the GM algorithm can be divided into two phases:

Initialization phase: in the first iterations $\left\|\underline{P} - \underline{P}^*\right\| \gg 0$ and $C_1^{GM} \gg 0$, therefore the convergence rate is linear.

Convergence phase: near the solution $\left\|\underline{P} - \underline{P}^*\right\| \to 0$, $C_1^{GM} \to 0$, and the convergence rate is quadratic according to C_2^{GM}, where C_2^{GM} is a function of the image properties.

An example of the verification as described above is demonstrated by registering the two images shown in FIG. 3. FIG. 3 shows a test of the Gauss-Newton convergence: (a) is an original "Airfield" image, and (b) is the "Airfield" image rotated by 30^0 using bilinear interpolation. An **X** mark indicates the initial estimate of the motion given as translation. The registration results are presented in FIG. 4, which indicates that the convergence process is divided into two phases: the first is related to a large motion estimation characterized by a low convergence rate $m_{(C1)}$, while the second is related to a small motion estimation characterized by a high convergence rate $m_{(C2)}$. The low-rate convergence corresponds to the linear convergence from n=0 to about n=170, and the high rate convergence corresponds to the quadratic convergence phase encountered after a cross-over point located at about n=170.

Multi-resolution GM scheme

The relationship between the pyramidal GM scheme and the convergence properties of the GM was studied by P. J. Burt, R. Hingorani and R J Kolczynski,

"Mechanisms for isolating component patterns in the sequential analysis of multiple motion", IEEE Workshop on Visual Motion, Princeton, New Jersey October 1991 (hereinafter BUR91) in the frequency domain for pure translations. By examining the error associated with the translation coefficients, the multi-scale scheme was proved to decrease the error term in Eq. 25 and improve the convergence rate.

Denote by $\underline{\varepsilon_{trans}}(s)$ - the error associated with the translation parameters (dx_s,dy_s) at a resolution scale s:

$$\underline{\varepsilon_{trans}}(s) = \begin{bmatrix} dx_s - dx_s^* \\ dy_s - dy_s^* \end{bmatrix} \tag{26}$$

where dx_s^* and dy_s^* are the optimal values of the translation parameters in scale s. Then, by scaling down the images from scale s_1 to scale s_2 ($s_2 > s_1$) one gets:

$$\begin{aligned} dx_{s_2} &= dx_{s_1} \cdot \frac{s_1}{s_2} \\ dy_{s_2} &= dy_{s_1} \cdot \frac{s_1}{s_2} \end{aligned} \tag{27}$$

and the associated error becomes

$$\underline{\varepsilon_{trans}}(s_2)^2 = \begin{pmatrix} dx_{s_2} - dx_{s_2}^* \\ dy_{s_2} - dy_{s_2}^* \end{pmatrix}^2 = \begin{pmatrix} dx_{s_1} \cdot \frac{s_1}{s_2} - dx_{s_2}^* \cdot \frac{s_1}{s_2} \\ dy_{s_1} \cdot \frac{s_1}{s_2} - dy_{s_2}^* \cdot \frac{s_1}{s_2} \end{pmatrix}^2 = \underline{\varepsilon_{trans}}(s_1)^2 \underbrace{\left(\frac{s_1}{s_2}\right)^2}_{<1}. \tag{28}$$

Hence, the error associated with the translation error is decreased by a factor of $\frac{s_1}{s_2}<1$.

Inserting Eq. 28 into Eq. 25 we get:

$$\begin{aligned} \varepsilon_{GM}\left(x_i^{(2)}, y_i^{(2)}, [dx, dy], s_2\right) &= \frac{1}{2} \sum_{P_i \in [dx,dy]} \frac{\partial^2 I_2(x_i^{(2)}, y_i^{(2)})}{\partial P_i^2} \underline{\varepsilon_{trans}}(s_2)^2 \\ &= \frac{1}{2} \sum_{P_i \in [dx,dy]} \frac{\partial^2 I_2(x_i^{(2)}, y_i^{(2)})}{\partial P_i^2} \left(\underline{\varepsilon_{trans}}(s_1)^2 \left(\frac{s_1}{s_2}\right)^2 \right) \\ &= \varepsilon_{GM}\left(x_i^{(2)}, y_i^{(2)}, [dx, dy], s_1\right) \underbrace{\left(\frac{s_1}{s_2}\right)^2}_{<1} \end{aligned} \tag{29}$$

Hence, for a dyadic pyramid, the truncation error of the Taylor approximation, related to the translation parameters (Eq. 29) is decreased by a factor of 4 in each increase of the pyramid's scale. Note that the error associated with motion parameters such as scale and shear, which are scale-invariant, is not reduced. Hence, multi-resolution schemes do not improve the convergence properties when the motion is dominated by scale and shear. This method can achieve higher convergence rate if instead of dyadic division, we use bigger scale factors.

Dominant motion locking

BUR91 used a frequency analysis to show that the coarse-to-fine refinement process allows the GM to lock on a single dominant motion even when multiple motions are present. This property is essential for most applications, which are based on image registration (SZE94, IRA93 and IRA95). The method presented above is utilized herein to provide an optimization-based analysis of this property, by studying the error associated with objects that perform dominant and non-dominant motions. This analysis extends the results of BUR91 by being applicable to general motion models - parametric and non-parametric.

Notation:

S The set of pixels that are common to I_1 and I_2

S_{Dom} A subset of S. This set of pixels that are common to I_1 and I_2, whose motion is the *dominant motion*, was defined above

S_{NonD} A subset of S. This set of pixels that are common to I_1 and I_2, whose motion is not the *dominant motion*

By permuting the rows of Eq. 13 according to the i^{th} pixel's relation, which belongs to either S_{Dom} or S_{NonD}, one gets:

$$\underline{\underline{\tilde{H}}}\,\underline{P} = \begin{bmatrix} \underline{\underline{\tilde{H}}}_{Dom} \\ \underline{\underline{\tilde{H}}}_{NonD} \end{bmatrix} \underline{P} = \begin{bmatrix} \underline{I}_t^{Dom} \\ \underline{I}_t^{NonD} \end{bmatrix} = \underline{\tilde{I}}_t \tag{30}$$

where

$$\underline{\underline{\tilde{H}}}_{Dom}\,\underline{P} = I_t^{Dom} \tag{31}$$

are the equations related to dominant motion, and

$$\tilde{\underline{\underline{H}}}_{NonD}\ \underline{P} = \underline{I}_t^{NonD} \tag{32}$$

are the equations related to non-dominant motion. As the GM algorithm converges to the dominant motion, the term $\underline{I}_t^{NonD}$ becomes a difference of uncorrelated pixels. Therefore, by assuming image isotropy, $\underline{I}_t^{NonD}$ can be modeled as uniformly distributed random variable with zero mean

$$E\{\underline{I}_t^{NonD}\} = 0. \tag{33}$$

This is a *landslide* type phenomenon: as the iterative solution $\underline{P}_n$ gets closer to the dominant motion's true parameters $\underline{P}^{Dom}$, the non-dominant pixels become more and more uncorrelated. Inserting Eq. 33 into Eq. 15:

$$\begin{aligned}\underline{E\{\Delta P\}} &= E\{\left(\tilde{\underline{\underline{H}}}^t\tilde{\underline{\underline{H}}}\right)^{-1}\tilde{\underline{\underline{H}}}^t\tilde{\underline{I}}_t\} \\ &= E\{\left(\tilde{\underline{\underline{H}}}^t\tilde{\underline{\underline{H}}}\right)^{-1}\tilde{\underline{\underline{H}}}^t\begin{bmatrix}\underline{I}_t^{Dom}\\ \underline{0}\end{bmatrix}\} + \underbrace{E\{\left(\tilde{\underline{\underline{H}}}^t\tilde{\underline{\underline{H}}}\right)^{-1}\tilde{\underline{\underline{H}}}^t\begin{bmatrix}\underline{0}\\ \underline{I}_t^{NonD}\end{bmatrix}\}}_{=0} \\ &= \underline{P}^{Dom}\end{aligned} \tag{34}$$

Thus, the non-dominant outliers are automatically rejected. The conclusion is that the GM is a non-biased estimator of the dominant motion parameters $\underline{P}^{Dom}$, where the variance of the estimation $Var(\underline{P}^{Dom})$ depends on the ratio between dominant and non-dominant pixels.

In order to better understand the method of the present invention, reference is first made to the general steps of the GM as described in FIG 5. An input step **100** to the algorithm includes providing two images $I_1(s)$, $I_2(s)$ (where s is the index of the current iteration of the algorithm) and an initial estimate of a geometric warping T, such that $T(I_2) \approx I_1$. When s=0, $I_1(0)$ and $I_2(0)$ are the input images to the process. The images are typically given as 2-D arrays, and the warping T is typically given as a parameters vector $\underline{P}$. The input images are smoothed in step **100**, typically using a separable averaging filter (application of an one-dimensional filter in the x direction and then in the y direction) whose 1-D component is [1/n, ... ,1/n], where n is the width of the filter. The resulting output images $I_1(s)$ and $I_2(s)$ (s=0) of step **100** are of

the same dimension as the source images. The output images are fed to a downscaling step **102.** In step **102** the images are downscaled by typically a factor of 3, using for example the following process: the images are smoothed typically using a separable kernel [1/3, 1/3, 1/3] and the parameters vector $\underline{P}$ is downscaled using the following equation:

$$\underline{\underline{P(s+1)}} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \underline{\underline{P(s)}} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{pmatrix} \quad (35)$$

The resulting output images are denoted $I_1(s+1)$, $I_2(s+1)$ and $\underline{P}(s+1)$

The downscaled image size is checked in a checking step **104**. If the image dimensions are not too small (minimal dimension of typically 20 pixels) the images and the parameters vector are downscaled again (return to step **102**). If the images are too small, the algorithm proceeds to an up-scaling step **106** in which higher resolution scale images $I_1(s+1)$ and $I_2(s+1)$ and an initial guess $\underline{P}_n(s+1)$ are output as $I_1(s)$, $I_2(s)$ and $\underline{P}_n$, where n is an index set initially to zero.

In a warping step **108**, $I_2(s)$ is warped towards $I_1(s)$ using $\underline{P}_n$. The result (output) is I_2^{warp}. Next, in a stopping step **110,** two stopping criteria are checked. If either one of them is satisfied, the algorithm proceeds to a step **112**, which checks the current resolution scale s: if this is the base resolution (s=0), the execution of the algorithm is terminated. If not, the algorithm returns to **106**. If neither of the stopping criteria in step **110** is met, the matrix $\underline{\underline{H}}^t\underline{\underline{H}}$ and vector $\underline{\underline{H}}^t\underline{I}_t$ are next calculated in a calculation step **114** using the following equations:

$$\left(H^T H\right)_{i,j} = \frac{1}{2}\sum_{k=1}^{m} \frac{\partial I_2^{warp}(k)}{\partial p_i} \frac{\partial I_2^{warp}(k)}{\partial p_j} \quad (36)$$

$$\left(\underline{\underline{H}}^T \underline{I}_t\right)_k - \sum_{j=1}^{m} \left(I_1^{warp}(k) - I_2^{warp}(k)\right) g \frac{\partial I_2^{warp}(k)}{\partial p_j} \quad (37)$$

where $I_2^{warp}(k)$ is the k^{th} pixel common to I_2^{warp} and $I_1(s)$. The outputs $\underline{\underline{H}}^t\underline{\underline{H}}$ and $\underline{\underline{H}}^t\underline{I}_t$ are then input to the equation $(\underline{\underline{H}}^t\underline{\underline{H}})\,\underline{\Delta P} = \underline{\underline{H}}^t\underline{I}_t$, which is solved in a solution step **116,** and the result $\underline{\Delta P}$ is used to update the estimate of the motion parameters vector $\underline{P}_{n+1}$ in an updating step **118**. The vector $\underline{P}_{n+1}$ is the sought after parameters information.

Major disadvantages of the basic GM in all its forms include the facts that it takes too long, consumes a lot of memory, and, in many cases, it does not converge. Therefore, the basic GM is not useful for practical video processing. Since GM is a fundamental key component in any video compression schemes, its poor performance prevents its use in video applications when computation speed is critical.

There is thus a widely recognized need for, and it would be highly advantageous to have, a method for fast motion estimation that does not suffer from the disadvantages of the basic GM listed above, and would therefore be useful in video applications.

SUMMARY OF THE INVENTION

The method of the present invention, as embodied by the bi-directional gradient method (BDGM) algorithm and the symmetric gradient method (SGM) algorithm represents a significant improvement over the Gradient Methods (GM) algorithm, which is considered to be the state-of-the-art algorithm for image registration, being able to estimate complicated motions at high sub-pixel accuracy. The BDGM provides a mechanism in which the algorithm finds the optimal location (in the sense that the errors of the calculated parameters are minimal) in the parameter space $\underline{P}$, while the SGM provides a mathematical formulation, which is symmetric regarding I_1 and I_2. Both approaches allow better convergence rates and a significantly improved convergence range, while using the same order of computational complexity. The BDGM and SGM are suited for video compression and creation of mosaics in video coding applications, such as the one defined by the MPEG4 video compression standard. As with the GM algorithm, the BDGM and SGM algorithms estimate the parameters vector $\underline{P}$ associated with the *parametric image registration* problem, which consists of estimating the following: pure global translation (2 parameters), image plane rotation (4 parameters), 2-D affine (6 parameters), and pseudo-projective (8-parameter flow). However, in contrast with the GM, the BDGM disclosed in the present invention estimates the parameters vector $\underline{P}$ by approaching a location in the range of each parameter which minimizes the error of the calculated parameters, while the SGM estimates symmetrically the center value of the value range of each parameter. A value range is referred to hereafter as an

"interval". This is an essential, innovative feature of the present invention that results in the superior convergence properties of the BDGM and SGM over GM in various motion estimation applications.

According to the present invention there is provided a method for fast global motion estimation, the global motion defined by a plurality of parameters in which each parameter has an interval, the method comprising providing a first and a second image, providing an initial estimate of each of two translation parameters, and determining the relative global motion between the first and second images using a symmetric gradient approach in an iterative process starting with the initial estimates of the two translation parameters, whereby the symmetric gradient approach provides the center of each the parameter interval and results in improved global motion estimation convergence properties.

According to preferred features in the symmetric gradient method of the present invention, the step of providing an initial estimate of each of two translation parameters includes providing an initial interval for each translation parameter and dividing each translation parameter initial interval into two equal sub-intervals, and the step of determining the relative global motion between the first and second images using a symmetric gradient approach includes providing a basic SGM formulation that includes the sub-intervals, running in each iteration a point-wise linearization procedure on the basic SGM formulation, and deriving in each iteration a symmetric linearization error based on the linearization procedure.

According to the present invention there is provided a method for fast global motion estimation, the global motion defined by a plurality of parameters in which each parameter has an interval, the method comprising providing a first and a second image, providing an initial estimate of each of two translation parameters, and determining the relative global motion between the first and second images using a bi-directional gradient approach in an iterative process starting with the initial estimates of the two translation parameters, whereby the bi-directional gradient approach provides the optimal location of each parameter interval, and results in improved global motion estimation convergence properties.

According to preferred features in the bi-directional gradient method of the present invention, the step of providing an initial estimate of each of two translation parameters includes providing an initial interval for each translation parameter and dividing each translation parameter initial interval into twosub-

intervals, and the step of determining the relative global motion between the first and second images using a bi-directional gradient approach includes providing a basic BDGM formulation that includes the sub-intervals, running in each iteration a point-wise linearization procedure on the basic BDGM formulation, and deriving in each iteration a bi-directional linearization error based on the linearization procedure.

The symmetric and bi-directional gradient approaches provide improved convergence properties to the motion estimation. The improved convergence properties include faster convergence than GM, or convergence in cases where the GM algorithms do not converge.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is herein described, by way of example only, with reference to the accompanying drawings, wherein:

FIG. 1 shows an image translation, in which two images have a common support area;

FIG. 2 shows a block diagram of the basic and iterative GM formulations. For n=0, $\underline{P}_0$ is given as an initial guess and $\underline{\Delta P}$ is the iterative update after each iteration;

FIG. 3 shows X marks after the application of the Gauss-Newton convergence in a typical GM algorithm on an original and a rotated image;

FIG. 4 shows the two phases of a typical GM convergence process;

FIG 5 shows a detailed block diagram of a typical gradient based motion estimation (GM) algorithm;

FIG. 6 shows graphically an 1-D illustration of a) basic GM techniques, b) the SGM technique, and c) the BDGM technique;

FIG. 7 shows block diagrams of replacement modules: a) SGM, and b) BDGM;

FIG. 8 shows registration results of rotated images using GM, SGM and BDGM: (a) 10^0 rotation, and (b) 30^0 rotation;

FIG. 9 shows test images for affine registration with small motion, where X marks the initial motion estimation;

FIG. 10 shows comparisons of the performance of the GM, SGM and

BDGM regarding the small motion registration of the images in FIG. 9;

FIG. 11 shows two images used to test the registration under poor illumination conditions, in which X marks the initial motion estimation;

FIG. 12 shows a comparison of the performance of GM, SGM and BDGM in the registration of the images of FIG. 11;

FIG. 13 shows two panoramic images with large motion, in which X marks the initial motion estimation;

FIG. 14 shows the registration results for the large panoramic motion for the images in FIG. 13, using GM, SGM, and BDGM;

FIG. 15 shows two panoramic images with small motion, in which X marks the initial motion estimation;

FIG. 16 shows the registration results for the small panoramic motion of the images in FIG. 15 using GM, SGM, and BDGM;

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention is of a method for fast global motion estimation using a bi-gradient or a symmetric gradient scheme. Specifically, the method of the present invention estimates the parameters vector $\underline{P}$ by providing either a bi-directional approach to a location inside the interval, or a symmetric approach to the center value of the interval of each parameter. The BDGM or SGM can be used to obtain faster GM convergence, or convergence in cases where the GM method does not converge.

The principles and operation of a fast global motion estimation method using bi-directional or symmetric gradient schemes according to the present invention may be better understood with reference to the drawings and the accompanying description.

The method of the present invention as embodied in its two main embodiments represents a significant improvement over the GM. The invention described herein discloses in detail a procedure to estimate the optimal or center value of each interval of each of the parameters of the vector $\underline{P}$ such that the error in the estimated parameters is minimal. The global motion estimation algorithm gets as an input an estimate (which is as an initial guess) of the two translation parameters Δx and Δy from the eight parameters it has to estimate in an iterative procedure in the

most general case. The other six motion parameters are initially assumed to be 0. No other information is required about the relative motion between the two images. In the BDGM formulation, the input images are used to approximate a virtual image that is situated optimally in the middle (not necessarily the center) of the parameters space P, defining the geometrical transformation $I_2 \rightarrow I_1$. In the SGM formulation the input images are used to approximate a virtual image that is situated in the center of the parameters space P, defining the geometrical transformation $I_2 \rightarrow I_1$. In other words, the BDGM algorithm is provided an initial guess of the intervals Δx and Δy, and proceeds in an iterative procedure to estimate the rest of the six parameters which lie in the interval of the parameters vector $\underline{P}$. In contrast, the SGM algorithm is provided with an initial guess of the intervals and starts the procedure by dividing the two initial values into two equal subsections. After the completion of the first iteration that provides an initial interval for each of the other (up to six) parameters of $\underline{P}$, the BDGM algorithm chooses points that lie in the interval of each of these estimated initial values of the parameters of $\underline{P}$, while the SGM algorithm takes the center of all the parameter values. This is the initial guess for the second iteration. After the completion of the second iteration, the BDGM algorithm chooses the middle (not necessarily the center) while the SGM algorithm takes the exact center of the newly (second iteration) estimated parameters values, and starts all over again.

This idea is easy to illustrate in the case of translation. FIG. 6 shows an illustration given for simplification purposes only in a 1-D space (only translation in the x direction is estimated), of the key difference in the technique of approaching the common pixels between I_1 and I_2: (a) in regular prior art GM, pixels in I_1 are approximated by pixels in I_2 over the (1-D) interval Δx; (b) in SGM pixels in the center of the interval between I_1 and I_2 are approximated by pixels common to I_1 and I_2; while in (c) BDGM, pixels between I_1 and I_2 are approximated by pixels common to I_1 and I_2, i.e. the original interval is divided into two sub-intervals $\underline{\Delta x_1}$ and $\underline{\Delta x_2}$. A meeting point **800** in (b) and (c) is chosen optimally to minimize the approximation error and to speed-up the computation. A similar approach is used for two-dimensional (2-D) space to estimate 2-D translation, rotation, 2-D affine and pseudo-projective movements. The BDGM or SGM estimate respectively the internal or center value of each interval for each parameter as done in the 1-D case. In other

words, the BDGM or SGM uses the common pixels to I_1 and I_2 in order to estimate the parameters that describe some internal location (BDGM) or the center (SGM) of the general translation, rotation, affine and pseudo-projective movements.

The key innovative feature of the SGM of the present invention, i.e. the symmetrical approach to the center value of each parameter interval, is shown schematically as a "replacement module" **119** in the block diagrams of FIG. 7a. Replacement module **119** is comprised of two novel steps **120** and **122.** These steps facilitate a symmetric approach to motion estimation based on GM. Steps **120** and **122** replace steps **114** and **116** of the GM in FIG. 5. In step **122**, the matrix $\left(\underline{\underline{H^t H}}\right)$ is calculated symmetrically and separately for I_2 and I_1, unlike the calculation in step **114** of the GM method. Thus, replacement module **119** of the SGM replaces the single iteration module or phase **30** in FIG. 2. The rest of the GM algorithm remains unchanged.

The key innovative feature of the BDGM of the present invention, i.e. that the algorithm automatically approaches an optimal location inside each parameter interval (minimizes the estimation error), is shown schematically as a "replacement module" **119'** in the block diagrams of FIG. 7b. Replacement module **119'** is comprised of three key new and novel steps **120', 124** and **126**. Steps **120'** and **124** replace step **114** of the GM in FIG. 5. Step **126** replaces step **116** in FIG. 5. In steps **120'** and **124**, the matrix $\left(\underline{\underline{H^t H}}\right)$ is calculated bi-directionally and separately for I_2 and I_1, unlike the calculation in step **114** of the GM method. Step **126** combines two parameter vectors into a single one, unlike step **114** in FIG. 5. Similar to the SGM, replacement module **119'** of the BDGM replaces only the single iteration module or phase **30** in FIG. 2. The rest of the GM algorithm remains unchanged**.** In order to better understand the innovative features of both SGM and BDGM, reference is first made to some key regular GM features.

The governing equations that describe the physical aspects of the system that governs the global motion are non-linear. Thus, one cannot find a fast solver that produces the right 2 parameters for pure global translation, 4 parameters for image plane rotation, 6 parameters for 2-D affine motion, and 8 parameters for pseudo-

projective motion. An efficient strategy to overcome the non-linear aspect of the problem uses linearization of the system of equations.

The basic GM formulation is given by Eqs. (38)-(41).

$$I\left(x_i^{(1)}\right) = I\left(x_i^{(2)}\right) \tag{38}$$

where $x_i^{(1)}$ and $x_i^{(2)}$ are the current estimates of the coordinate of the i^{th} sample common to I_1 and I_2 satisfying:

$$x_i^{(1)} = x_i^{(2)} + \Delta x \tag{39}$$

Eq. 38 is solved by expanding I_2 in a first order Taylor expansion and solving for Δx :

$$I\left(x_i^{(1)}\right) = I\left(x_i^{(2)}\right) + \frac{\partial I_2\left(x_i^{(2)}\right)}{\partial x} \cdot \Delta x \,. \tag{40}$$

This point-wise expansion in the linearization causes an error estimated by Eq. 25 :

$$\varepsilon_{GM} = \varepsilon\left(x_i^{(2)}, \Delta x\right) = \frac{1}{2}\frac{\partial^2 I_2(\tilde{x}^{(2)})}{\partial x^2} \cdot \Delta x_i^2, \quad \tilde{x} \in [0, \Delta x] \tag{41}$$

Both the BDGM and the SGM of the present invention substantially improve the performance of the GM by estimating the optimal locations (BDGM) or center (SGM) of the parameters vector. In the 1-D case, Δx is the interval where the computation is taking place, and the GM error (Eq. 41) has a quadratic behavior and is proportional to Δx^2. A main innovative feature of the BDGM and SGM of the present invention is in providing a way to decrease this error. Both the fact that there should be a decrease, and the way to achieve such a decrease are non-obvious, as detailed and demonstrated hereinbelow.

Referring again to the 1-D case for the sake of simplicity, the error is decreased according to the present invention by dividing the 1-D interval into sections or sub-intervals. The sections may be equal or non-equal in size. Specifically, the algorithm either yields automatically the location in the 1-D interval in which the algorithm operates (BDGM), or a-priori divides it into two equal sections (SGM), and reduces the error by a factor of 2 as shown below. In general, the error is proportional to $\frac{\Delta x^2}{n}$, where n is the number of sections into which the original interval is divided. The same approach is used when rotation, 2-D affine and pseudo-

projective movements are present. For example, if a true rotation includes a rotation of 30° between two images, the SGM will estimate this rotation by dividing it into two parts: from 0 to 15° and from 15° to 30°. The SGM estimates the correct exact center value of each interval for each parameter of the parameters vector describing the particular motion (each of the four parameters for rotation in the example above). In contrast, the BDGM will not start with an a-priori division of the rotation into two equal parts, but will run an iterative optimization that yields the optimum (not necessarily center) location of each rotation parameter value interval.

The solver of the present invention is based on an iterative algorithm. In the regular GM methodology, as the number of iterations increases, the error is reduced. As stated and shown below, by using the BDGM or SGM of the present invention the error can be reduced by factor of 2 when the problem is solved in two half sub-intervals. Thus, the same error as in the regular GM can be reached in the iterative procedure of either embodiment of the method of the present invention much faster.

In order to more appropriately distinguish between the two main embodiments of the method of the present invention, i.e. the SGM and BDGM, the two are presented separately below.

SGM

In the 1-D case, in order to reduce the estimation error, and therefore improves the convergence, the estimation error $\varepsilon_{GM}\left(x_i^{(2)},\Delta x\right)$ of Eq. 41 is reformulated symmetrically in respect to Δx according to FIG. 6(b), to give a basic SGM formulation:

$$I_1\left(x_i^{(1)}-\Delta x/2\right)=I_2\left(x_i^{(2)}+\Delta x/2\right) \tag{42}$$

Expanding both sides using Taylor expansion:

$$I_1\left(x_i^{(1)}\right)-\frac{\partial I_1\left(x_i^{(1)}\right)}{\partial x}\cdot\frac{\Delta x}{2}+\varepsilon_{GM}\left(x_i^{(1)},\Delta x/2\right)=I_2\left(x_i^{(2)}\right)+\frac{\partial I_2\left(x_i^{(2)}\right)}{\partial x}\cdot\frac{\Delta x}{2}+\varepsilon_{GM}\left(x_i^{(2)},\Delta x/2\right) \tag{43}$$

$$\underbrace{I_2\left(x_i^{(2)}\right)-I_1\left(x_i^{(1)}\right)}_{-I_t}+\left(\frac{\partial I_1\left(x_i^{(1)}\right)}{\partial x}+\frac{\partial I_2\left(x_i^{(2)}\right)}{\partial x}\right)\cdot\frac{\Delta x}{2}+\underbrace{\varepsilon_{GM}\left(x_i^{(1)},\Delta x/2\right)-\varepsilon_{GM}\left(x_i^{(2)},\Delta x/2\right)}_{\varepsilon_{GM}\left(x_i^{(2)},\Delta x\right)}=0 \tag{44}$$

Next, an upper bound is derived for a symmetric linearization error associated with Eq. 43:

$$\varepsilon_{SGM}\left(x_i^{(2)},\Delta x\right)\le 2\varepsilon_{GM}\left(x_i^{(2)},\Delta x/2\right) \tag{45}$$

The error term is quadratic in ΔX. Then by comparing ε_{SGM} to the regular GM error ε_{GM} of Eq. 41:

$$\varepsilon_{GM}\left(x_i^{(2)},\Delta x/2\right)=\frac{1}{4}\varepsilon_{GM}\left(x_i^{(2)},\Delta x\right) \tag{46}$$

$$\begin{aligned}\varepsilon_{SGM}\left(x_i^{(2)},\Delta x\right)&=2\varepsilon_{GM}\left(x_i^{(2)},\Delta x/2\right)\\&=\frac{1}{2}\varepsilon_{GM}\left(x_i^{(2)},\Delta x\right).\end{aligned} \tag{47}$$

A better approximation error analysis can be derived by expanding $\varepsilon_{SGM}\left(x_i^{(2)},\Delta X\right)$:

$$\begin{aligned}\varepsilon_{GM}\left(x_i^{(2)},\Delta x/2\right)-\varepsilon_{GM}\left(x_i^{(1)},\Delta x/2\right)&=\frac{1}{4}\varepsilon_{GM}\left(x_i^{(2)},\Delta x\right)-\frac{1}{4}\varepsilon_{GM}\left(x_i^{(1)},\Delta x\right)\\&=\frac{1}{8}\Delta x_i^2\left(\frac{\partial^2 I_2\left(\tilde{x}_i^{(2)}\right)}{\partial x^2}-\frac{\partial^2 I_1\left(\tilde{x}_i^{(1)}\right)}{\partial x^2}\right)\\&\approx\frac{1}{8}\Delta x_i^2\left(\frac{\partial^3 I_2\left(\tilde{x}\right)}{\partial x^3}\Delta x\right)\\&=\frac{\partial^3 I_2\left(\tilde{x}\right)}{\partial x^3}\frac{\Delta x^3}{8}\end{aligned} \tag{48}$$

The smaller the symmetric linearization error ε_{SGM}, the better is the convergence rate. If ε_{SGM}=0, Eq. 40 converges in a single iteration.

The 1-D case above was treated in detail as an example only, in order to demonstrate the main ideas of the invention. The 2-D "general" cases, which include translation, rotation, affine and pseudo-projective movements are treated next. The general approach is more difficult to visually demonstrate that the simple 1-D approach of FIG. 6b. In the general 2-D case, the "center" of each parameter in the

parameters vector $\underline{P}$ is computed iteratively to a final value, and this reduces the estimation error and therefore improves substantially the convergence rate.

In the general 2-D case, the SGM is formulated using the motion parameters vector $\underline{P}$

$$\underline{P}^* = \arg\min_{\underline{P}} \left\{ \sum_{(x_i,y_i)\in s} \left(I_2\left(x_i^{(1)}, y_i^{(1)}, \tfrac{\underline{P}}{2}\right) - I_2\left(x_i^{(2)}, y_i^{(2)}, -\tfrac{\underline{P}}{2}\right)\right)^2 \right\} \tag{49}$$

$$r_i\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right) = I_2\left(x_i^{(1)}, y_i^{(1)}, \tfrac{\underline{P}}{2}\right) - I_1\left(x_i^{(2)}, y_i^{(2)}, -\tfrac{\underline{P}}{2}\right) \tag{50}$$

From Eq. 50 one has

$$\frac{\partial r_i\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right)}{\partial \underline{P}} = \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}, \tfrac{\underline{P}}{2}\right)}{\partial \underline{P}} - \frac{\partial I_1\left(x_i^{(2)}, y_i^{(2)}, -\tfrac{\underline{P}}{2}\right)}{\partial \underline{P}} \tag{51}$$

Using the chain rule $\dfrac{\partial I_1\left(x_i^{(1)}, y_i^{(1)}, -\tfrac{\underline{P}}{2}\right)}{\partial \underline{P}}$ is expressed in terms of $\dfrac{\partial I_1\left(x_i^{(1)}, y\right)}{\partial \underline{P}}$:

$$\frac{\partial I_1\left(x_i^{(1)}, y_i^{(1)}, \tfrac{\underline{P}}{2}\right)}{\partial \underline{P}} = \frac{\partial I_1\left(x_i^{(1)}, y_i^{(1)}, \tfrac{\underline{P}}{2}\right)}{\partial\left(\tfrac{\underline{P}}{2}\right)} = \frac{1}{2}\frac{\partial I_1\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right)}{\partial \underline{P}}. \tag{52}$$

Therefore, one has

$$\frac{\partial r_i\left(x_i^{(1)}, y_i^{(1)}, \tfrac{\underline{P}}{2}\right)}{\partial \underline{P}} = \frac{1}{2}\left(\frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}, \tfrac{\underline{P}}{2}\right)}{\partial \underline{P}} + \frac{\partial I_1\left(x_i^{(1)}, y_i^{(1)}, \tfrac{\underline{P}}{2}\right)}{\partial \underline{P}} \right) \tag{53}$$

Assuming

$$\frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}, \tfrac{\underline{P}}{2}\right)}{\partial \underline{P}} \approx \frac{\partial I_1\left(x_i^{(1)}, y_i^{(1)}, \tfrac{\underline{P}}{2}\right)}{\partial \underline{P}} \tag{54}$$

one gets

$$\frac{\partial r_i\left(x_i^{(1)}, y_i^{(1)}, \tfrac{\underline{P}}{2}\right)}{\partial \underline{P}} \approx \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}, \tfrac{\underline{P}}{2}\right)}{\partial \underline{P}} \tag{55}$$

Taking the second derivative and using Eq. 54

$$\frac{\partial^2 r_i\left(x_i^{(1)}, y_i^{(1)}, \underline{P}/2\right)}{\partial \underline{P}} = \frac{1}{2}\left(\frac{1}{2}\frac{\partial^2 I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial \underline{P}^2} - \frac{1}{2}\frac{\partial^2 I_1\left(x_i^{(1)}, y_i^{(1)}\right)}{\partial \underline{P}^2}\right) \approx 0. \tag{56}$$

Comparing Eq. 56 to Eqs. 18 and 23

$$C_1^{SGM} = \frac{C_1^{GM}}{2} \tag{57}$$

$$C_2^{SGM} = C_2^{GM} \tag{58}$$

Therefore, the SGM will generally outperform the regular GM algorithm in the convergence rate due to its reduced linearization error.

5 SGM algorithm flow

The matrix $\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)$ is in general calculated separately as respectively a first matrix for I_1 and a second matrix for I_2 in matrix calculation step **120**, using preferably the following formulations

$$\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)_{k,j}^{I_1} = \sum_i \frac{\partial I_1\left(x_i^{(1)}, y_1^i\right)}{\partial P_k} \frac{\partial I_i\left(x_i^{(1)}, y_2^i\right)}{\partial P_j} \tag{59}$$

$$\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)_{k,j}^{I_2} = \sum_i \frac{\partial I_2\left(x_i^{(2)}, y_2^i\right)}{\partial P_k} \frac{\partial I_2\left(x_i^{(2)}, y_2^i\right)}{\partial P_j} \tag{60}$$

Following step **118** in FIG. 7a

$$\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)^{SGM} \underline{P}^{SGM} = \underline{\underline{H}}^t \underline{I_t} \tag{61}$$

where a combined matrix $\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)^{SGM}$ is preferably given by

$$\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)_{k,j}^{SGM} = \frac{1}{2}\left(\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)_{k,j}^{I_1} + \left(\underline{\underline{H}}^t \underline{\underline{H}}\right)_{k,j}^{I_2}\right) \tag{62}$$

and $\left(\underline{\underline{H}}^t \underline{I_t}\right)$ is preferably calculated according to Eq. 17.

The SGM returns $\underline{P}^{SGM}$ as the result denoted as $\underline{\Delta P}$ in FIG. 7a.

BDGM

In the 1-D case of FIG. 6c, in order to reduce the estimation error by decreasing the linearization error in the interval $[0 \ldots \Delta X]$, the algorithm is allowed to partition optimally this interval using the following formulation

$$I\left(x_i^{(1)} - \Delta x_1\right) = I\left(x_i^{(2)} - \Delta x_2\right) \tag{63}$$

Using Eq. 41 and a Taylor expansion of Eq. 63

$$I_1\left(x_i^{(1)}\right) - \frac{\partial I_1\left(x_i^{(1)}\right)}{\partial x} \cdot \Delta x_1 + \varepsilon_{GM}\left(x_i^{(1)}, \Delta x_1\right) = I_2\left(x_i^{(2)}\right) + \frac{\partial I_2\left(x_i^{(2)}\right)}{\partial x} \cdot \Delta x_2 + \varepsilon_{GM}\left(x_i^{(2)}, \Delta x_2\right) \tag{64}$$

$$\frac{\partial r_i\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right)}{\partial \underline{P}} = \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}, \frac{P}{2}\right)}{\partial \underline{P}} - \frac{\partial I_1\left(x_i^{(2)}, y_i^{(2)}, -\frac{P}{2}\right)}{\partial \underline{P}} \tag{65}$$

where the motion between I_2 and I_1 is given by:

$$\Delta x = \Delta x_1 + \Delta x_2 \tag{66}$$

We analyze the error term of Eq. 65:

$$\begin{aligned} 2 \cdot \varepsilon_{BDGM}\left(x_i^{(1)}, x_i^{(2)}, \Delta x_1, \Delta x_2\right) &= 2\left(\varepsilon_{GM}\left(x_i^{(2)}, \Delta x_2\right) - \varepsilon_{GM}\left(x_i^{(1)}, \Delta x_1\right)\right) \\ &= \frac{\partial^2 I_2\left(\tilde{x}^{(2)}\right)}{\partial x^2}\Delta x_2^2 - \frac{\partial^2 I_1\left(\tilde{x}^{(1)}\right)}{\partial x^2}\Delta x_1^2 \\ &= \frac{\partial^2 I_2\left(\tilde{x}^{(2)}\right)}{\partial x^2}\Delta x_2^2 \underbrace{- \frac{\partial^2 I_2\left(\tilde{x}^{(2)}\right)}{\partial x^2}\Delta x_1^2 + \frac{\partial^2 I_2\left(\tilde{x}^{(2)}\right)}{\partial x^2}\Delta x_1^2}_{0} - \frac{\partial^2 I_1\left(\tilde{x}^{(1)}\right)}{\partial x^2}\Delta x_1^2 \\ &= \frac{\partial^2 I_2\left(\tilde{x}^{(2)}\right)}{\partial x^2}\left(\Delta x_2^2 - \Delta x_1^2\right) + \Delta x_1^2\left(\frac{\partial^2 I_2\left(\tilde{x}^{(2)}\right)}{\partial x^2} - \frac{\partial^2 I_1\left(\tilde{x}^{(1)}\right)}{\partial x^2}\right) \\ &= \frac{\partial^2 I_2\left(\tilde{x}^{(2)}\right)}{\partial x^2}\left(\Delta x_2 - \Delta x_1\right)\left(\Delta x_2 + \Delta x_1\right) + \Delta x_1^2\left(\frac{\partial^3 I_2\left(\tilde{x}\right)}{\partial x^3}\Delta x\right) \end{aligned} \tag{67}$$

Substituting Eq. 66 into Eq. 67:

$$\varepsilon_{BDGM}\left(x_i^{(1)}, x_i^{(2)}, \Delta x_1, \Delta x_2\right) = \frac{\Delta x}{2}\left(\frac{\partial^2 I_2\left(\tilde{x}^{(2)}\right)}{\partial x^2} \cdot \left(\Delta x_2 - \Delta x_1\right) + \Delta x_1^2 \frac{\partial^3 I_2\left(\tilde{x}\right)}{\partial x^3}\right) \tag{68}$$

and ε_{BDGM} is the error of the bi-directional GM. Since the solution of Eq. 65 minimizes ε_{BDGM} directly, we expect to achieve superior convergence results. For the symmetric case where $\Delta x_1 = \Delta x_2 = \frac{\Delta x}{2}$, the first term of Eq. 68 vanishes and

$$\varepsilon_{BDGM}\left(x_i^{(1)}, x_i^{(2)}, \Delta x_1, \Delta x_2\right) = \frac{\partial^3 I_2(\tilde{x})}{\partial x^3}\frac{\Delta x^3}{8} \tag{69}$$

An extension into two dimensions with general motion models is given next for the BDGM algorithm.

As in the SGM, the 1-D BDGM case was treated in detail as an example only, in order to demonstrate the main ideas of the invention. In the general 2-D case, the location of each parameter in the parameters vector $\underline{P}$ is computed iteratively to a final value, and this reduces the estimation error and therefore improves substantially the convergence rate.

In the general 2-D case, the BDGM is formulated using the motion parameters vector $\underline{P}$

$$\underline{P}^* = \arg\min_{\underline{P}}\left\{\sum_{(x_i,y_i)\in s}\left(I_2\left(x_i^{(2)}, y_i^{(2)}, \underline{P}_2\right) - I_1\left(x_i^{(1)}, y_i^{(1)}, -\underline{P}_1\right)\right)^2\right\} \tag{70}$$

where $\underline{P}_1$ and $\underline{P}_2$ have the same dimensions as the motion parameters vector used in the GM formulations. The overall motion is given by

$$\underline{P}=\underline{P}_1+\underline{P}_2. \tag{71}$$

Let $k, m \in [0\ldots1], k+m=1$, then:

$$\begin{aligned}\underline{P}_1&=k\cdot\underline{P}\\ \underline{P}_2&=m\cdot\underline{P}\end{aligned} \tag{72}$$

$$\begin{aligned} r_i\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right) &= I_2\left(x_i^{(1)}, y_i^{(1)}, \underline{P}_2\right) - I_1\left(x_i^{(2)}, y_i^{(2)}, -\underline{P}_1\right)\\ &= I_2\left(x_i^{(1)}, y_i^{(1)}, k\underline{P}\right) - I_1\left(x_i^{(2)}, y_i^{(2)}, -m\underline{P}\right)\end{aligned} \tag{73}$$

$$\frac{\partial r_i\left(x_i^{(1)},y_i^{(1)},\underline{P}\right)}{\partial \underline{P}}=\frac{\partial I_2\left(x_i^{(2)},y_i^{(2)},k\underline{P}\right)}{\partial \underline{P}}-\frac{\partial I_1\left(x_i^{(1)},y_i^{(1)},-m\underline{P}\right)}{\partial \underline{P}}$$
$$=k\frac{\partial I_2\left(x_i^{(2)},y_i^{(2)}\right)}{\partial \underline{P}}+m\frac{\partial I_1\left(x_i^{(1)},y_i^{(1)}\right)}{\partial \underline{P}} \tag{74}$$

$$\frac{\partial^2 r_i\left(x_i^{(1)},y_i^{(1)},\underline{P}\right)}{\partial \underline{P}^2}=k^2\frac{\partial^2 I_2\left(x_i^{(2)},y_i^{(2)}\right)}{\partial \underline{P}^2}+m^2\frac{\partial^2 I_1\left(x_i^{(1)},y_i^{(1)}\right)}{\partial \underline{P}^2} \tag{75}$$

By assuming symmetry one gets

$$\frac{\partial^2 r_i\left(x_i^{(1)},y_i^{(1)},\underline{P}\right)}{\partial \underline{P}^2}=\left(k^2+m^2\right)\frac{\partial^2 I_2\left(x_i^{(2)},y_i^{(2)}\right)}{\partial \underline{P}^2}$$
$$=\left(k^2+\left(k-1\right)^2\right)\frac{\partial^2 I_2\left(x_i^{(2)},y_i^{(2)}\right)}{\partial \underline{P}^2} \tag{76}$$
$$=\left(2k-1\right)\frac{\partial^2 I_2\left(x_i^{(2)},y_i^{(2)}\right)}{\partial \underline{P}^2}$$

Thus

$$C_2^{BDGM}=C_2^{GM} \tag{77}$$

5 And the optimal partitioning of the interval $[0\ldots\underline{P}]$, which minimizes $\left|\frac{\partial^2 r\left(x_i^{(1)},y_i^{(1)},\underline{P}\right)}{\partial \underline{P}^2}\right|$, is the symmetric approach $\left(k=\frac{1}{2}\right)$. Furthermore, the partitioning used by the regular GM is worse than the bi-directional formulation, since $\left|\frac{\partial^2 r\left(x_i^{(1)},y_i^{(1)},\underline{P}\right)}{\partial \underline{P}^2}\right|$ is maximized for k=0, m=1 or k=1, m=0.

BDGM algorithm flow

The matrix $\underline{\underline{(H^t H)}}$ is in general calculated separately as respectively a first matrix for I_1 and a second matrix for I_2 in matrix calculation step **120'** in FIG. 7b, using preferably the following formulations

$$\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)^{I_1}_{k,j} = \sum_i \frac{\partial I_1\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_k} \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_j} \tag{78}$$

$$\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)^{I_2}_{k,j} = \sum_i \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_k} \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_j} \tag{79}$$

Let

$$\underline{\underline{H}}^{BDGM} \stackrel{\Delta}{=} \left[\underline{\underline{H}}^{I_1} \quad \underline{\underline{H}}^{I_2}\right] \tag{80}$$

$\underline{\underline{H}}^{BDGM}$ is a matrix of dimensions $(n_{pixels} \times 2 \cdot n_{param})$, where n_{param} is the number of motion parameters and n_{pixels} is the number of pixels common to I_2 and I_1.

1. Denote by $\underline{P}^{BDGM}$ the BDGM parameters vector. Then

$$\underline{P}^{BDGM} = \begin{bmatrix} \underline{P}^1 \\ \underline{P}^2 \end{bmatrix} \tag{81}$$

$\underline{P}^1$ and $\underline{P}^2$ are vectors of dimension $(n_{param} \times 1)$.

2. The solution of the equation in step **126** of FIG. 7b is

$$\left(\left(\underline{\underline{H}}^{BDGM}\right)^t \underline{\underline{H}}^{BDGM}\right) \underline{P}^{BDGM} = \left(\underline{\underline{H}}^{BDGM}\right)^t \underline{I}_t \tag{82}$$

where $\underline{I}_t$ is similar to the one used in the Basic GM. After solving Eq. 82, the solution $\underline{P}^{BDGM}$ is given by:

$$\underline{P}^{BDGM} = \underline{P}^1 + \underline{P}^2 \tag{83}$$

EXPERIMENTAL RESULTS

Exemplary performances of the SGM and BDGM embodiments of the method of the present invention in various motion estimation problems, and a comparison with GM results obtained in the same problems are given next. The numerical results are expressed in terms of alignment error vs. the number of iterations needed for

convergence. Real and simulated image pairs were used. The same implementations of the iterative refinement and multi-scale embedding were used for the SGM, BDGM and GM algorithms. Thus, the only difference is in the bi-directional approach in BDGM and the symmetric approach to a "center point" of parameter intervals in SGM. The multi-scale decomposition is preferably constructed using a three-tap filter [1/3, 1/3, 1/3] and the derivative is preferably approximated using [½ 0 -½], for example by following M. Elad, P. Teo, Y. Hel-Or, "Optimal Filters For Gradient-Based Motion Estimation", Technical Report # 111, HP Lab, 1997, and E. P. Simoncelli, "Design of multi-dimensional derivatives filters", IEEE International Conf. on Image Processing, Austin, Texas, 1994. Other filters may be also used. The initial estimate is an estimate of the translation parameters. The SGM and BDGM are tested by estimating large and small motions using several motion models: rotation, affine and pseudo-projective. "Small" rotation motion is defined as a rotation of typically less than about 10 degrees. "Small" translation is defined as typically less than 10 pixels in the lowest resolution. Conversely, "large" rotation motion is defined as equal to or more than 10 degrees, and "large" translation is defined as typically more than 10 pixels in the lowest resolution. "Very large" rotations are rotations of typically more than about 30 degrees.

Estimation of large and very large rotations

The estimation of large and very large rotations is illustrated using the image of FIG. 3. The image is rotated using a bilinear interpolation, while the background areas created by the rotation are padded with zeros. The registration is calculated using a linearized rotation model:

$$\begin{aligned} x_1 &= a \cdot x_2 - b \cdot y_2 + c \\ y_1 &= b \cdot x_2 + a \cdot y_2 + f \end{aligned} \tag{84}$$

Figure 8(a) shows the performance of registering an image rotated by 10^0, which is considered to be a large rotation. The BDGM converges twice as fast as the GM: 4 iterations compared to 7 iterations. The SGM converges in 5 iterations. Both the BDGM and the SGM significantly outperform the GM by converging in 25 iterations compared to the GM's 37 iterations. This means that we get the same accuracy with

fewer iterations. In other words, the same results are achieved much faster than with the regular GM.

Estimation of small affine motion

According to Eqs. 57 and 77 respectively, the SGM and BDGM are expected to perform similarly to the GM when registering small motions. In order to verify this experimentally, the two images in FIG. 9 (a, b) were registered using the affine motion model:

$$\begin{aligned} x_1 &= a \cdot x_2 + b \cdot y_2 + c \\ y_1 &= d \cdot x_2 + e \cdot y_2 + f \end{aligned} \tag{85}$$

As in FIG. 3, the X mark indicates the common location where the computations take place. The results of estimation of small affine motion obtained with both GM and BDGM using the same conditions are presented in FIG. 10. The GM, SGM and BDGM algorithms convergence is similar, occurring after 4-5 iterations.

Registration of images with low contrast

Instability in the registration process can also be attributed to images that have low contrast. In this type of images, the spatial derivatives are very small

$$\begin{aligned} &\frac{\partial r\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right)}{\partial \underline{P}} \to 0 \quad \frac{\partial r\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right)}{\partial \underline{P}} \to 0 \\ &\frac{\partial^2 r\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right)}{\partial \underline{P}^2} \to 0 \quad \frac{\partial^2 r\left(x_i^{(1)}, y_i^{(1)}, \underline{P}\right)}{\partial \underline{P}^2} \to 0 \end{aligned} \tag{86}$$

Under such conditions, according to Eqs. 23 and 24, the convergence rate of the GM deteriorates. The two images presented in FIG. 11(a, b) are real airborne images, which are registered using the affine motion model defined by Eq. 85. As in FIGS. 3 and 9, the X mark indicates common locations between the images where the computation starts. The convergence results, presented in FIG. 12, show that the SGM is able to converge to the solution (after ca. 26 iterations), while the GM completely diverges. There is no numerical instability of the SGM in the proximity of the solution, in contrast to the divergence of the GM. Thus, the SGM yields clearly superior results and enables registration of images with low contrast, a registration

that is not possible with prior art GM. The BDGM is also able to converge to the solution, although it oscillates before its convergence.

Estimation of large and very large panoramic motion

The registration of panoramic images is of special importance, since it is the basis for most mosaic-based applications discussed in the Background section. The motion model used for panoramic image registration is the pseudo-projective model (MAN94 and SZE94):

$$x_1 = \frac{a \cdot x_2 + b \cdot y_2 + c}{g \cdot x_2 + h \cdot y_2 + 1}$$
$$y_1 = \frac{d \cdot x_2 + e \cdot y_2 + f}{g \cdot x_2 + h \cdot y_2 + 1} \qquad (87)$$

Due to large number of unknowns and the non-linear nature of Eq. 87, the GM based registration becomes slow and unstable. Two sets of images photographed by a regular 35mm cameras were used to compare between the performance of the GM and SGM registration algorithms: FIG. 13 for a large panoramic motion and FIG. 15 for a small panoramic motion. The X mark in each indicates the common location where the computations start.

The registration results for the large panoramic motion, shown in FIG. 14, demonstrate the superior convergence of the BDGM (13 iterations) and SGM (17 iterations) compared to the GM (23 iterations). The registration results for the small panoramic motion, shown in FIG. 16, also demonstrate superior convergence of the SGM and BDGM, and show that both the SGM and the BDGM converge almost twice as fast (5 iterations) as the GM (9 iterations).

In summary, the method of the present invention in its two main preferred embodiments includes new formulations and algorithms that enhance the performance of gradient-based image registration methods. These algorithms extend the current state-of-the-art image registration algorithms, and are shown to possess superior convergence range and rate. By analyzing the convergence properties using non-linear optimization algorithms, explicit expressions are derived for the convergence of the

GM. The experimental results verify the theoretical analysis. The improved convergence rate of the SGM and BDGM are especially vital for advanced video compression standards such as MPEG-2, 4, H26*, etc, particularly when implemented on low-power mobile devices.

All publications, patents and patent applications mentioned in this specification are herein incorporated in their entirety by reference into the specification, to the same extent as if each individual publication, patent or patent application was specifically and individually indicated to be incorporated herein by reference. In addition, citation or identification of any reference in this application shall not be construed as an admission that such reference is available as prior art to the present invention.

While the invention has been described with respect to a limited number of embodiments, it will be appreciated that many variations, modifications and other applications of the invention may be made.